Redes de bits como alternativa às redes neurais em problemas de aprendizado por reforço

Nickolas R. Machado [1], Pedro C. F. Machado[1], Juliana M N S Zamith[1], Filipe Braida[1], Leandro Alvim[1], Raul S. Ferreira[2]

[1]Departamento de Ciência da Computação – Universidade Federal Rural do Rio de Janeiro (UFRRJ) - Rio de Janeiro – RJ – Brasil

[2]Departamento de Computação e Robótica – LAAS-CNRS, Toulouse, França

Resumo. *Redes neurais artificiais (RNAs) são amplamente utilizadas, como por exemplo, em jogos digitais através de agentes inteligentes que replicam o comportamento humano. Contudo, o custo computacional de treinamento delas costuma ser alto, exigindo a escolha entre maior processamento e menor qualidade. Desta forma, este trabalho propõe a aplicação de redes de bits (RBs) como uma alternativa capaz de maximizar o processamento e minimizar o uso de memória. Comparados às RNAs de precisão simples, os resultados mostram um speedup de até 91 vezes usando 32 vezes menos memória.*

1. Introdução e Motivação

Redes neurais são aplicadas com sucesso em problemas da computação. Contudo, o bom desempenho delas exige alto custo computacional, limitando a utilização pelo *hardware*, principalmente em arquiteturas embutidas. Entre as soluções para este problema estão as redes de *bits* (*RB*), uma alternativa que consome menos memória e faz uso de menos instruções que redes de precisão simples (*RF32*) [1][2].

Isso se deve às *RBs* substituírem os parâmetros das redes neurais (pesos), que geralmente são números reais em palavras de 32 *bits (Float)* por *bits*, o que reduz o espaço necessário para representar a informação, uma vez que um *Float* pode constituir 32 pesos e não apenas um. Assim, o uso de *RBs* pode diminuir em até 32x a quantidade de memória utilizada. Ainda, a representação de *bits* permite trocar vários procedimentos de multiplicação por uma operação *bit* a *bit* do tipo *XOR* [1].

Outro fator relevante é o impacto no treinamento das *RBs.* Tendo em vista que o espaço de busca de uma *RF32* é de aproximadamente $(2^{32})^p$ combinações, onde p é o número de pesos, em redes com quantificação binária, isto é, as que assumem os pesos apenas como valores 0 ou 1 representando respectivamente +1 e -1, esse monte se reduz a 2^p [1]. Logo, o tempo gasto para encontrar uma das soluções é menor.

Contudo, a retirada dos números reais leva à perda de precisão da rede, podendo ser necessário ter mais neurônios e/ou aplicá-la apenas a algumas classes de problemas onde não é necessária alta exatidão na entrada. Um exemplo disto é o jogo *Snake.*

Este trabalho tem como objetivo estudar e verificar o impacto da quantificação binária, na entrada e nos pesos, e o uso de *bits* no tempo de execução e no consumo de memória durante o andamento da aplicação. Além disso, dois cenários de testes foram implementados para uma análise comparativa entre *RBs* e *RF32s:* um deles retrata o

desempenho das redes com entradas e pesos aleatórios, e outro com elas aplicadas ao treinamento no *Snake*. A próxima seção apresenta detalhes da implementação destes casos, bem como um algoritmo genético [3] utilizado no treinamento.

2. Implementação do jogo *Snake* utilizando redes de *bits*

Na implementação do *Snake* deste trabalho, o usuário, neste caso uma rede neural, deve mover a cobra pelo cenário em busca da comida. Ainda, quando a cabeça da cobra (*head*) encontra um alimento, o tamanho dela é acrescido e a pontuação aumentada, e ao encostar em uma parte do corpo ou em uma parede, há derrota. Além disso, a posição inicial dos alimentos foi fixada para criar um contexto controlado de treinamento. Outrossim, a cobra trata-se de um agente que possui 20 sensores fornecendo dados binários, do tipo está e não está, calculados com relação às posições da *head* e das 4 direções (norte, sul, oeste, leste), localizando a comida, partes do corpo e as paredes.

A *RB* e *RF32* utilizadas possuem 20 neurônios na camada oculta, escolhidos para que a rede possa elaborar de estratégias complexas no jogo, e 4 na camada de saída para os possíveis movimentos da cobra. Ademais, a implementação foi feita em *Julia* por ser uma linguagem com características de alto nível e foco em desempenho. Por fim, foram usados vetores do tipo *UInt64* nos pesos das *RBs* que possibilitam o armazenamento e execução de 64 *bits* através de uma única operação.

Para o treino das redes foi feita uma versão do algoritmo genético com seleção elitista, *crossover* com corte em ponto, mutação de *bias* e pesos e avaliação [3] baseada nas comidas coletadas, distância do alimento e movimentos que o afastam deste último.

3. Testes

Os testes descritos nesta seção foram realizados em um *Core i7 8750h Intel* de 6 núcleos e 12 *threads* e *16GB* de memória *DDR4*. O primeiro teste consistiu em avaliar a performance da *RB*. Para tanto, foi usado um vetor de valores aleatórios como entrada de dados e 512 neurônios com pesos também aleatórios. Ainda, foram implementadas três versões do código: a primeira delas, chamada V_{Bit} , que utiliza *RBs*; a segunda variante, V_{FL32} , constitui-se de forma similar à V_{Bit} , mas operando com *RF32s;* e por fim a V_{Flux}, que foi implementada empregando o *Flux*[1], uma biblioteca para *machine learning* desenvolvida em *Julia*.

A Tabela 1 apresenta o tempo de execução e a quantidade de memória consumida nas três versões. Variou-se o tamanho da entrada para avaliar o impacto desta no desempenho. Com isso, é visto através da tabela que o consumo médio aproximado de memória da rede de *bits* é 32x menor quando comparada com os outros dois casos. Quanto ao tempo de execução, verificando as variantes V_{Bit} e V_{FL32} foi obtido um *speedup* de até 91x nos testes. Contudo, quando comparadas as versões V_{Bit} e V_{Flux} o maior *speedup* médio foi em torno de 10x. Acredita-se que esta diferença decorre das otimizações proporcionadas pelo *BLAS*[2], uma biblioteca para álgebra linear com diversos aprimoramentos matemáticos e de uso dos recursos de *hardware*, empregada pelo *Flux* na execução das *RF32s*.

[1] Flux, https://fluxml.ai/ .
[2] BLAS, http://www.netlib.org/blas/

Tabela 1. Quantidade de memória utilizada, tempo de execução e *speedup* obtidos para uma média de 600 execuções.

Entrada	Tempo (µs)			Tamanho (MB)			*Speedup*	
	V_{Bit}	V_{FL32}	V_{Flux}	V_{Bit}	V_{FL32}	V_{Flux}	$V_{FL32\ x\ Bit}$	$V_{Flux\ x\ Bit}$
512	5.542	283.343	14.761	0.031	1.000	1.002	51.126	2.663
4096	31.646	2402.852	123.680	0.250	8.000	8.002	75.929	3.908
16384	105.387	9667.600	1059.619	**1.000**	**32.000**	**32.002**	**91.734**	**10.054**

O segundo teste consistiu em avaliar o desempenho das *RBs* no treinamento aplicado ao *Snake*. Os resultados obtidos são apresentados no Gráfico 1. Neste, é possível perceber uma considerável redução no número de gerações obtidas na versão que usa *RBs* (V_{Bit}) comparado com a outra que utiliza *RF32s* (V_{Flux}). Contudo, o tempo não diminuiu como esperado. Isto se deve à computação da avaliação não ser o mais custoso no treinamento, e a mutação de *bits* requerer mais operações para ser realizada.

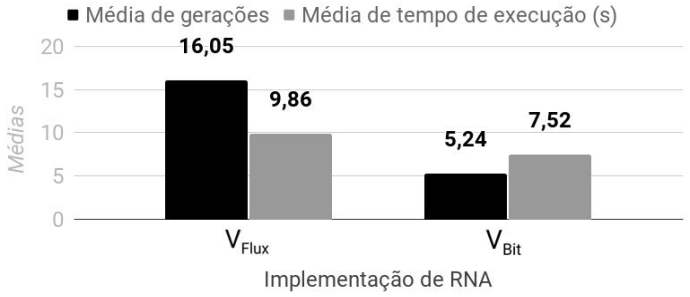

Gráfico 1. Comparação média de 100 execuções entre redes de bits e Float32 na convergência do jogo Snake até atingir o placar mínimo de 63 pontos na avaliação.

5. Conclusão

Através dos testes é possível perceber que a execução em *RBs* é mais rápida que em *RF32s*, enquanto nem sempre essa melhora é expressiva no treinamento. Contudo, dependendo do uso dessas redes na avaliação e o impacto dos pesos binários no treinamento, o *speedup* neste pode ser um fator decisivo para usá-las. Ainda, mesmo que haja um treinamento demorado, o consumo de memória e o desempenho das *RBs* oportuniza o emprego delas em *CPU*, viabilizando uma gama de usos para elas.

6. Referências

[1] Rastegari, M., Ordonez, V. , Redmon, J. and Farhadi, A. (2016) "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks" https://arxiv.org/pdf/1603.05279.pdf , Agosto.

[2] Assis, P. (2009) "O que são Redes Neurais?" https://bit.ly/3mqTBN2 , Setembro.

[3] Massago, S. (2013) "Introdução ao Algoritmo Genético" https://cutt.ly/GhfdpJL , Agosto.